O protocolo MOS e o SIGDiC

Jorge Bertocchini[1],João Paulo Rodrigues[1]

INESC Porto, R. Dr. Roberto Frias, 4200-465 Porto
http://www.inescporto.pt/
jbertocchini@netcabo.pt,jpr@inescporto.pt

Resumo O MOS (Media Object Server Protocol) é um protocolo criado para comunicação entre aplicações de redacção - Newsroom Computer Systems (NCS) - e equipamentos de Broadcast - Media Object Servers (MOS)-, como sejam, servidores de vídeo ou áudio, arquivos multimédia, geradores de caracteres, entre outros equipamentos.
Pela sua importância actual em ambientes profissionais de broadcast, o protocolo MOS foi escolhido, não só para a integração do Sistema Integrado de Gestão e Difusão de Conteúdos (SIGDiC) com os restantes equipamentos profissionais, mas também para implementação de mecanismos de comunicação interna entre os vários módulos do SIGDiC.
A natureza distribuída do SIGDiC sobre uma rede MOS, implicava a necessidade de implementar elementos que terminariam o protocolo MOS em cada módulo, o que levou ao desenvolvimento de uma framework em Java.

1 Introdução

O trabalho aqui apresentado foi realizado no âmbito de um projecto financiado pelo PRIME - Sistema Integrado de Gestão e Difusão de Conteúdos (SIGDiC)[5] - cujo objectivo é desenvolver um sistema capaz de integrar diferentes fontes de conteúdos e disponibilizá-los, automaticamente ou semi-automaticamente, em diferentes canais de comunicação (SMS, Teletexto, WAP, Web, etc.).

Tendo em vista a definição de requisitos, o projecto tem como parceiro uma entidade líder em Portugal na área da produção, edição, gestão e difusão de conteúdos - a RTP. Contudo, o SIGDiC foi concebido tendo em mente as necessidades da generalidade das organizações que desenvolvem o seu "core-business"nesta área e, deste modo, são potenciais utilizadores dos resultados do projecto, todas as empresas nacionais e internacionais que produzem, editam e geram conteúdos para os sectores do audiovisual, Internet e telecomunicações.

O SIGDiC irá ter uma arquitectura modular, baseada em sistemas abertos, e está a ser desenvolvido recorrendo, tanto quanto possível, a ferramentas de domínio público. É neste contexto que surge a opção pelo MOS (Media Object Server Communications Protocol)[4], como tecnologia de middleware, pelo facto de ser um protocolo open-source e por, actualmente, constituir um standard de facto como protocolo de interligação entre equipamentos heterogéneos de múltiplos fabricantes.

O MOS é um protocolo emergente, construído para possibilitar uma comunicação fácil e simples dentro de sistemas de produção de notícias servindo, especialmente, para a interligação de sistemas de redacção - Newsroom Computer Systems (NCS) - e Media Object Servers, tais como servidores de vídeo e geradores de caracteres, entre outros.

Exemplos do que o MOS permite fazer:

1) Usando Media Servers e comunicação através de MOS, produtores em centros noticiosos podem gerir alinhamentos de programas com conteúdos multimédia em estações distantes.

2) Usando software MOS, é possível incluir conteúdos multimédia, simplesmente arrastando o apontador MOS para o corpo da notícia.

2 O Protocolo MOS

O protocolo MOS coordena a comunicação entre uma central NCS e uma série de servidores MOS especializados. Neste contexto, o protocolo abrange um modelo de objectos, onde o NCS possui referências para todos os conteúdos que estão incluídos nos Media Object Servers. À medida que um programa está a ser produzido e o seu alinhamento, que inclui todos os respectivos conteúdos, é preenchido, o NCS transfere as referências, como os alinhamentos, para cada MOS e recebe retorno acerca do status de cada peça.

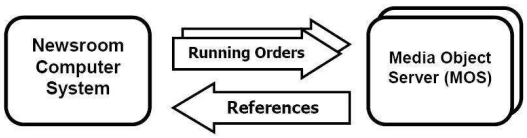

Figura 1. Interacção NCS - MOS (Tipos de Mensagens)

Dado que cada sistema gere a sua própria base de dados de objectos (conteúdos MOS, peças noticiosas, alinhamentos, elementos multimédia, entre outros), o protocolo troca referências e avisos entre as diferentes bases de dados quando um dos sistemas executa um evento (criação, destruição, agendamento ou controlo). O protocolo define um conjunto de notificações, que são enviadas numa ou noutra direcção durante a operação.

A versão 2.8 do protocolo MOS permite uma representação de metadata adicional (incluída no campo mosExternalMetadata), considerada como uma "caixa preta"dentro do protocolo MOS, ou seja, é guardada e trocada de uma forma transparente entre os intervenientes, mas a sua interpretação cai fora do âmbito do protocolo.

O MOS usa dois portos TCP/IP de forma bidireccional, onde as aplicações estão constantemente à escuta de mensagens. Estes portos, designados por MOS Upper Port(10541) e MOS Lower Port(10540), são utilizados para o envio de mensagens do NCS para o MOS e do MOS para o NCS, respectivamente.

Por exemplo, um comando de criação de um alinhamento ("running order") iniciado no NCS e o respectivo ACK, que será enviado pelo MOS, ocorrerão no MOS Upper Port, ao passo que comandos indicando actualizações de objectos localizados no MOS e o respectivo ACK enviado pelo NCS, tomarão lugar através do MOS Lower Port .

A partir versão 2.x, as messagens MOS são enviadas em XML[9], de acordo com as regras definidas no MOS Data Type Definition (DTD). Nas versões mais antigas da norma, 1.x, os campos de dados eram delimitados por um formato próprio.

No protocolo MOS todos os campos são case-sensitive e as mensagens deverão constituir documentos XML bem-formados, mas não é necessário que sejam válidos.

Cada mensagem MOS terá de ter uma raiz (<mos>), seguida dos identificadores do MOS e do NCS (¡mosID¿ e ¡ncsID¿), após o qual virá o tipo de mensagem. Os dados referentes a cada tipo de mensagem vêm em seguida, dentro de elementos XML.

```
<mos>
  <mosID>INESC.TICKER.MOS</mosID>
  <ncsID>LX1-ENPS</ncsID>
  <roStorySend>
     <roID>LX1-ENPS;P_TRAINING\W;56AE879D-DC9A-425C-8619C0324CA23C7B</roID>
     <storyID>LX1-ENPS;P_TRAINING\W\R_56AE879D-DC9A-425C-8619C( ... )D713B</storyID>
     <storySlug>Nova Story Insert</storySlug>
     <storyNum/>
     <storyBody>
        <p>teste de nova historia</p>
     </storyBody>
        <mosExternalMetadata>
        <mosScope>PLAYLIST</mosScope>
        <mosSchema>http://LX1-ENPS/schema/enps.dtd</mosSchema>
        <mosPayload>
           <Tema>Nacional</Tema>
           <Ticket>DIA DA MULHER - Jorge Sampaio condecorou 22 mulheres
                   portuguesas que se distinguiram nas artes e nas letras</Ticket>
           <Approved>0</Approved>
           <ModBy>JORGE</ModBy>
           <ModTime>20040422163917</ModTime>
           <Printed>20040308195450</Printed>
           <StoryProducer>ok</StoryProducer>
           <Owner>JORGE</Owner>
           <Creator>JORGE</Creator>
           <TextTime>10</TextTime>
           <MediaTime>140</MediaTime>
        </mosPayload>
     </mosExternalMetadata>
  </roStorySend>
</mos>
```

Figura 2. Mensagem MOS

O elemento referente à metadata é um dos mais versáteis e nele vão estar inseridas as informações definidas pelo utilizador, em função das necessidades para as quais a solução foi desenhada. Por exemplo, poderá incluir o corpo de

uma notícia, ao mesmo tempo que inclui informações acerca do método como essa notícia pode ser usada na página da Web e no portal WAP. Além disso, pode também conter informações quanto à forma como poderá ser exibida no teletexto, ou através de envio por SMS, desde o tipo de letra à cor utilizada. Não há limites ao tipo de informação que se pretenda colocar dentro desse(s) elementos(s), desde que sejam correctamente processados pelas aplicações que funcionam dentro dos serviços respectivos.

O conjunto de mensagens MOS, que foram divididos em perfis, definem um conjunto de funcionalidades suportadas. Dado que a implementação de todos os perfis não é obrigatória é comum, nas soluções comerciais, apenas uma parte dos perfis serem suportados.

Cada perfil implementa as seguintes funcionalidades:

- Perfil 0: Comunicações básicas
- Perfil 1: Fluxo básico orientado a objectos
- Perfil 2: Fluxo básico de conteúdos / alinhamentos
- Perfil 3: Fluxo avançado orientado a objectos
- Perfil 4: Fluxo avançado de conteúdos / alinhamentos
- Perfil 5: Controlo de elementos
- Perfil 6: Redireccionamento de MOS

Para uma empresa poder afirmar que o seu produto é compatível com MOS, o sistema deverá suportar, no mínimo o perfil 0 e um outro perfil adicional.

3 O SIGDiC e o MOS

O resultado final do projecto SIGDiC deve ser um sistema preparado para ser integrado facilmente numa arquitectura que promova a reutilização de conteúdos, bem como, a integração de processos de trabalho e fluxos de informação.

Baseado nesta premissa, e tendo em conta que o SIGDiC deverá estar preparado para ambientes profissionais que, cada vez mais, usam MOS para interligação entre equipamentos, o suporte para MOS passou a ser um requisito fundamental.

O MOS poderia ser utilizado como protocolo fronteira, permitindo a interligação do SIGDiC com os restantes sistemas proprietários do ambiente de redacção. Contudo, a opção tomada foi mais arrojada e consistiu em trazer o MOS para "dentro"do SIGDiC, utilizando-o como protocolo de interligação interno. Deste modo, a arquitectura SIGDiC assenta sobre uma rede MOS, em que cada elemento se apresenta como um nó dessa rede.

A opção pelo protocolo de comunicações MOS deve-se à enorme implantação que esta tecnologia tem no meio televisivo em particular e em todo o meio noticioso em geral. Actualmente, o MOS é utilizado em ambientes de redacção profissional como protocolo de comunicação entre sistemas heterogéneos, de múltiplos fabricantes. A opção por um protocolo de comunicação que se transformou num standard de facto na área de intervenção do projecto é, com certeza, a opção correcta, contrariamente à opção de definir e implementar um protocolo de comunicação específico e proprietário. Esta opção permite uma maior integração de sistemas, facilitando a sua utilização em ambientes profissionais.

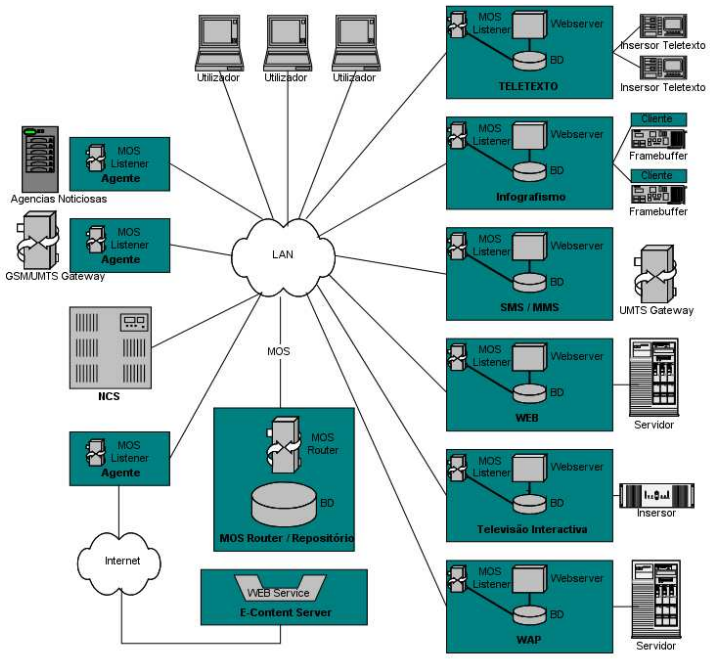

Figura 3. Arquitectura SIGiC

Uma vantagem adicional do protocolo MOS é ser inteiramente baseado em XML. Este aspecto permite passar os pacotes de informação, de forma transparente, através de qualquer rede de dados e, também, que este protocolo seja facilmente transformado num qualquer protocolo proprietário, recorrendo a transformações XML.

Na arquitectura do SIGDiC cada módulo tem funcionalidade própria, mesmo quando isolado. Contudo, quando integrado com os restantes módulos, obtêm-se uma funcionalidade estendida, pelo aumento da integração entre os vários módulos e a consequente automação de processos. Para comunicação entre os vários módulos, será utilizado o protocolo MOS, apresentando-se cada módulo na rede como um equipamento MOS. Assim, é possível operar cada módulo separado da restante arquitectura, como um sistema independente, que pode interoperar directamente com o restante equipamento de redacção.

Apesar de o protocolo MOS não ser um protocolo pensado para troca de conteúdos, a existência do elemento MosExternalMetadata, que não é analisado e que é transportado, transparentemente, nas mensagens, fornece um mecanismo que permite a troca transparente de informação. O SIGDiC explora esta característica para implementar o mecanismo de comunicação interno, que inclui a troca de conteúdos.

O repositório do SIGDiC assume o papel de concentrador de informação e, para além de recepcionar e enviar as mensagens MOS passadas entre módulos,

tem também a função de guardar as várias actualizações realizadas a um conteúdo. Por exemplo, a criação de uma notícia dá origem a uma mensagem MOS de notificação que, ao chegar ao concentrador, é reenviada, de forma automática, para todos os módulos que devam ser notificados e armazenada no repositório.

Por exemplo, a criação de uma mensagem a ser mostrada em rodapé, no módulo de Infografismo do SIGDiC, originaria uma mensagem MOS que, ao ser processada pelo concentrador, originaria um anexo ao conteúdo inicial, que passaria a ser constituído pela notícia e pela sua versão em rodapé.

O concentrador aparece assim como um "potenciador automático"da reutilização de conteúdos, ao manter um repositório permanentemente actualizado, com a informação que se encontra disseminada pelos vários módulos utilizados.

Cada módulo incorpora um componente que designamos por "MOS Listener", o qual pretende implementar o ponto terminal para o protocolo de comunicações. Este componente é sempre constituído por duas partes: 1) uma parte genérica de terminação de protocolo; 2) uma parte específica que implementa as funcionalidades próprias do módulo. De acordo com a mensagem recebida, cada módulo originará diferentes procedimentos, que tratarão a mesma informação de forma também diferente.

4 Implementação da Framework Java para MOS

A adopção do protocolo MOS, como protocolo de comunicação entre todos os módulos do SIGDiC, obriga a criar listeners específicos em cada módulo, que sejam capazes de terminar correctamente o protocolo MOS. Cada listener deverá realizar o processamento das mensagens MOS recebidas, de acordo com a função a desempenhar em cada módulo.

Com o objectivo de promover a reutilização de código fonte, tem vindo a ser desenvolvida uma framework Java para MOS. Esta framework irá implementar toda a parte de terminação do protocolo e fornecer mecanismos expeditos de criação de mensagens, deixando apenas para cada um dos diferentes módulos a implementação do tratamento específico das mensagens recebidas.

As mensagens MOS estão estruturadas em XML. Para podermos manipular facilmente os valores inseridos nos campos da mensagem, optou-se por se utilizar DOM, neste caso JDOM[2].

As principais classes que constituem a framework são:

1. `Start`
   Classe que implementa o método que dá início ao programa. Abre os portos usando as classes MOSUpperPort (para o porto 10541) e MOSLowerPort (para o porto 10540), de forma a receber ligações.
2. `MOSProtocolServer`
   Uma das classes fundamentais porque é a que implementa o protocolo a nível dos sockets. Implementa um servidor multi-thread que resulta na criação de uma nova thread para processar cada pedido. Sempre que o socket recebe um pedido de ligação, é instanciada uma classe do tipo MOSMessageHandler para processar a mensagem.

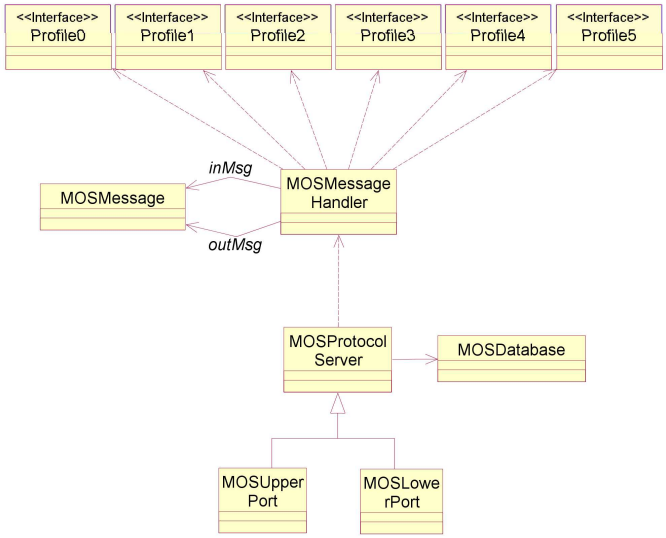

Figura 4. Framework Java

3. `MOSUpperPort,MOSLowerPort`
   Estas duas classes implementam uma especialização de MOSProtocolServer, implementando o serviço em cada um dos portos: MOS Upper Port(10541) e MOS Lower Port(10540).
4. `MOSDatabase`
   Classe utilitária que implementa uma camada de abstracção à base de dados. Implementa métodos utilitários e que encapsulam os métodos JDBC[7], promovendo abstracção de alguma complexidade do JDBC. A configuração da ligação JDBC é obtida, por sua vez, a partir de um ficheiro XML.
5. `Profile0 . . . Profile5`
   Interfaces Java para cada um dos perfis das mensagens MOS. Cada interface inclui a assinatura dos métodos que tratam cada uma das mensagens desse perfil. Qualquer solução deverá implementar as interfaces ProfileX, correspondentes aos perfis que são suportados.
6. `MOSMessageMOSMessage`
   Esta classe está directamente relacionada com o formato de cada mensagem. Inclui diversos construtores para a criação de novas mensagens e funções utilitárias para tratar mensagens, recebidas ou a enviar. A classe MOSMessage é uma subclasse da classe Document, da package JDOM, usada para manter uma representação numa árvore DOM das mensagens XML.
7. `MOSMessageHandler`
   Classe que processa as mensagens recebidas; Invoca o método adequado para o processamento da mensagem; Determina as respostas adequadas a cada mensagem.

Sempre que uma nova mensagem é recebida, o parser do XML é executado através da criação de um novo objecto do tipo MOSMessage. De acordo com o tipo de mensagem MOS recebida, é invocado o método correspondente que irá fazer o tratamento da mensagem, processar o pedido e enviar a resposta.

Em implementações normais de MOS, o programador, praticamente, só terá que implementar o processamento das mensagens recebidas, através da implementação dos métodos das interfaces correspondentes aos Perfis suportados.

5 Conclusões

O protocolo MOS foi desenvolvido por fabricantes, software houses e utilizadores com o objectivo de criar mecanismos que permitissem o controlo sobre os objectos multimédia disponíveis nos servidores de video e geradores de caracteres. Apesar do protocolo não ter sido alvo de um processo de normalização, a sua elevada implantação em meios profissionais de broadcasting, transformou-o num standard *de facto* para ambientes de redacção.

Apesar de não ser um protocolo para troca de conteúdos, o MOS suporta um mecanismo de transporte transparente de informação que pode ser utilizado para estender as funcionalidades do protocolo.

Tirando partido desta funcionalidade foi possível utilizar o protocolo MOS para implementar as comunicações inter-módulos do SIGDiC. Ao implementar cada módulo como um "produto"MOS, cada módulo pode interagir directamente com outros equipamentos profissionais.

Esta possibilidade foi já verificada através de testes em ambiente real, com o protótipo do SIGDiC. Verificou-se que a aplicação de Redacção utilizada na RTP[1], uma solução comercial proprietária, foi capaz de controlar o módulo de infografismo do SIGDiC. O sistema de redacção criou, automaticamente, a informação de noticias passadas no rodapé em programas noticiosos, que eram postas no ar pelo módulo de infografismo.

O desenvolvimento da framework Java, permitiu que o desenvolvimento das várias componentes MOS fosse realmente simplificado, traduzindo-se numa redução substancial do tempo de desenvolvimento.

Referências

[1] AP. Associated Press ENPS - MOS Connection , 2003. `http://www.enps.com/mos/`.

[2] JDOM. JDOM Project Homepage, 2004. `http://www.jdom.org/`.

[3] João Paulo Rodrigues Jorge Bertocchini. Implementação de um protocolo de comunicações baseado em MOS. Projecto/trabalho de fim de curso, FEUP / INESC Porto, 2004.

[4] MOS. MOS Protocol Homepage, 1999. `http://www.php.net`.

[5] João Paulo Rodrigues. SIGDiC - Sistema Integrado de Gestão e Difusão de Conteúdos. Tese de mestrado, INESC Porto, 2004.

[6] Alexandria SC. JavaServide - Java to NT Service Connector, 2003. `http://www.alexandriasc.com/software/JavaService/`.

[7] Sun. Sun JDBC Tutorial , 2002. http://java.sun.com/docs/books/tutorial/jdbc.
[8] W3C. XSL Transformations, 1999. http://www.w3.org/TR/xslt.
[9] W3C. Extensible Markup Language (XML), 2003. http://www.w3.org/XML/.